PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

CONCEPTS FOR A MODEL DRIVEN REQUIREMENTS BASED DEVELOPMENT PROCESS

Johannes Werner

Ilmenau University of Technology
Department System and Software Engineering
PO Box 10 05 65, 98684 Ilmenau, Germany
johannes.werner@tu-ilmenau.de

ABSTRACT

The development of complex embedded systems usually is carried out by application of requirements based processes. Such processes are characterised by analysis and breakdown of textual requirements specifying the desired behaviour and other characteristics of the new system. This fact is also the main weakness, because a plain textual description is not sufficient to obtain a complete and consistent image of a system.

In opposition to that, a completely model based process tries to describe the characteristics of a system within a formal model. The elaboration of such a formal model is carried out step by step, whereas the first steps contain a high level of abstraction. In the later steps this abstraction is continuously replaced by the detailed formal description of the system. Implying an adequate modelling environment, such a formal model can be used for simulation at any step of development. Thus validation and verification can be achieved much earlier. The main handicap of model based development lies in the increasing efforts in the early project stages, wherein also a lot of experience of the involved developers is mandatory.

This work tries to evolve concepts for a successful combination of both fundamental approaches. It shall be pointed out, that restructuring requirements analysis by means of SysML models results in increasing quality of specification. The application of UML models lower the risk of realising defective behaviour. Furthermore it will be shown, that virtual integration in an executable cumulative model can be done for validating and verifying the systems' complex behaviour.

Index Terms— Model, Requirements, Systems Engineering Processes, Unified Modelling Language (UML), Systems Modelling Language (SysML)

1. INTRODUCTION

Within the realisation of complex embedded systems not only experienced developers, but also high-quality development processes are needed.

Today the most common approach is a requirements based development process (requirements based systems engineering, RBSE) as for instance described in [1]. This type of process has been proven and tested in embedded systems engineering for several decades and very often lead to more or less successful products' development.

During the last years not only the complexity of embedded systems rose significantly, but also the degree of interconnectedness (for example see [2], [3] and [4]). As pointed out by [2], the use of textual requirements involves the risk of describing the systems' complex behaviour inadequately. According to [5], the main error source is the inaccuracy of the used language itself, followed by truly defective logic and content. Even assuming a quite perfect specification of a single system, there might still be incorrect interaction with adjacent systems. This might be caused by errors in the plain text interface description or by violation of non-functional requirements, which hardly can be revealed before realisation (see [2] and [3]).

As depicted by [5], one main approach for improvement is an increase of accuracy of language including a strict monitoring of this accuracy. For instance [2] and [4] suggest the usage of a graphical model notation to address that problem. Assuming the selection of a suitable graphical language, an execution of the resulting specification model might lead to early verification of the systems' behaviour and interaction. Following this argumentation, a strictly model based approach solves the problem of defective requirements and leads to early validation of complex interconnected systems (see [2] and [4]).

But despite this advantages there still is a lack of acceptance for such kind of model based systems engineering (MBSE). Basically decision-making stakeholders often shy away from increasing efforts at early project stages both in requirements and in model based engineering processes. But in model based engineering the main steps of creating an initial specification cannot be neglected like often done in requirements based development, according to [5]. Even if errors in an initial specification model can be detected and fixed more easily in model based compared to requirements based systems engineering processes, there is nevertheless the need for the best available expert knowledge. Often this modelling specific expert knowledge is missing because of the described lack of acceptance.

Another main problem related to a strict model centric process is the notion, that artefacts of previous development stages are implicitly included inside the current state of the evolving model. As for instance described by [4], this point of view is contrary to main aspects of traditional development processes. Especially in saftety critical systems development there usually exists a multitude of regulations. For example in the area of avionics or automotive systems the engineering process must provide textual specification documents for product certification purposes. This kind of plain text documentation is missing in solely model based systems engineering.

In consideration of the main pros and cons of the two development approaches, this paper tries to evolve basic concepts combining both methods. It will outline a model driven requirements based systems engineering process. As intro-

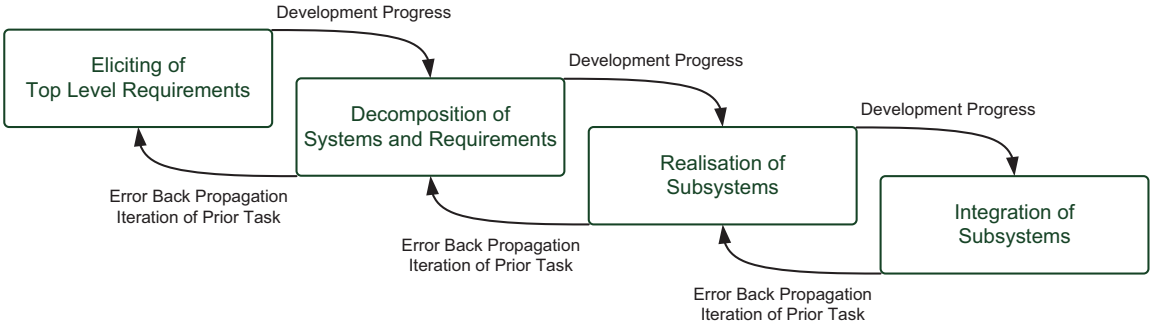

Fig. 1. Main steps of the described requirements based systems engineering process

duction to the topic, first this paper will give a short sketch of the two basic systems engineering methods. Afterwards the main part describes the key techniques for a promising combination of these methods. The last part tries to evaluate and judge the achieved results before an outlook closes the paper.

2. REQUIREMENTS BASED SYSTEMS ENGINEERING

Requirements based systems engineering processes, as for example described by [1], focus on the task of the so called requirements management. This task is a complex process of mainly eliciting, analysing, documenting and literally managing requirements which a system (or a product, a service, etc.) is expected to fulfil. The term requirements usually denotes the use of plain text specification, sometimes additionally illustrated by diagrams.

In the first step of RBSE top level requirements are gathered from targeted users, customers or other identified stakeholders. These requirements define the basic capabilities and characteristics of the intended system.

On basis of these general requirements the system is decomposed into smaller subsystems. Within this decomposition task every top level requirement has to be assigned to one of the resulting subsystems or, if this is not possible, the requirement also has to be decomposed accordingly. This requirements break down targets to obtaining smaller parts of the whole system, which are less complex and so easier to handle.

After a finite number of decomposition steps the subsystems have reached a manageable complexity and can now be passed on to expert teams for further processing. If the requirements of a subsystem are fine-grained enough at that point, the experts start realising the subsystem. Otherwise some more iterations of decomposition may lead to a suitable granularity of requirements.

After realisation the subsystem can now be verified and validated (subsystem level V&V). Afterwards step by step more complex systems are assembled from tested subsystems (integration). Hereafter also the integrated systems can get checked for errors and compliance to the specified characteristics (integration/system level V&V).

At the end of an also finite number of successful integration and V&V iterations the resulting overall system should meet the specified capabilities and characteristics.

It shall be pointed out, that not only the tasks of decomposition and integration are commonly iterated more than once. If at any step of requirements management inaccuracy or defectiveness of previous results are discovered, they have to be fixed by iteration of the prior tasks.

In the described requirements based development it is neccesary, that every step is precisely documented. Especially the linkage between higher level and decomposed requirements is essential. Particularly back propagation of errors depends on thoroughly managed requirements break down history.

For the inevitable realisation of this so called traceability also the literal management of requirements is important. As this is commonly done by usage of commercial tools (for example see [4]) and has no direct effect on the main development procedure, this task shall not be described any further here.

Figure 1 shows the main steps of the requirements based systems engineering process as described above.

As mentioned earlier, requirements based development is the most common approach in embedded systems engineering. Usually long trained expert knowledge and experience is available.

Another key advantage is the missing preliminary definition of subsequently used methods and tools. An elaborated specification can theoretically be realised with any imaginable domain specific technique. Even the tool support for requirements eliciting and decomposition is not mandatory but strongly recommended for preserving traceability.

However, this flexibility is also one of the key disadvantages. Plain text requirements can aid information exchange, because there is no need for learning a new language. But in the same way this can be a handicap. As described before, the missing accuracy leads to diverse problems. Especially incompleteness, contradictoriness or other inconsistencies are often not recognisable or either provable. This challenge intensifies with increasing interconnectedness and more stringent performance requirements.

3. MODEL BASED SYSTEMS ENGINEERING

To eliminate inaccuracy as the key disadvantage of requirements based systems engineering, commonly the introduction of model based development processes is suggested (for example see [2] and [4]).

The term model is applied in multiple disciplines like for example in mathematics and physics (e.g. mathematical models), in psychology (e.g. behavioural models), in computer science and engineering (e.g. architectural system models, functional system models). Due to this diversity of model usage and interpretation first a short definition of the notion of the term model within this paper shall be given.

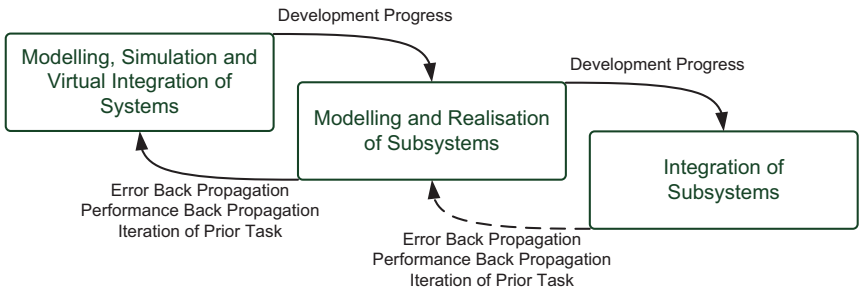

Fig. 2. Main steps of the described model based systems engineering process

As the commonly accepted general definition by [6] points out, models are mainly characterized by three features:

1. Representation: A model is a representation of an archetype, which again can be a model.
2. Concentration: Only those aspects and properties of the archetype are transfered to the model, that are needed for analysis.
3. Pragmatism: The type and detailedness of the model is basically determined by the intention of analysis. For example static and dynamic models represent such different pragmatic model types.

Briefly summing up, models are abstracting descriptions of the system for a predefined analysis goal. So models of one and the same system can differ entirely due to their purpose.

According to [2], complex interconnected systems can only be verified and validated entirely by simulation within mission level[1] scenarios. Following that argumentation, in the notion of this paper only dynamic models lead to a truly model based systems engineering.

As denoted previously, even requirements based processes try to take advantage of clarifying diagrams. Often this diagrams use standardised modelling languages like the Unified Modelling Language (UML, see [8]), the Systems Modelling Language (SysML, see [9]) or the Integrated Definition Language (IDEF, see [10]). But in solely requirements based systems engineering these diagrams are only used as static illustrations and so not as models in the notion of this paper.

In compliance with this definitions and the cited ressources, model based systems engineering focuses on establishing and evolving dynamic models. By simulation of those models the complex behaviour and performance parameters of the systems can be analysed theoretically at any stage of development.

At the beginning of the engineering process a highly abstracted model is generated. On basis of this so called black box model an iterative hierarchical refinement takes place. Within this refinement the black box parts of the system model get elaborated using available information about internal functionality and performance. This leads to grey and later white box models. As previously described, the amount of applied information mainly depends on the intention of analysis.

Similar to the requirements based development, parts of the model can be passed on to expert teams for further processing.

[1] for clarification of the term mission level see [7]

If after some iterations a system or subsystem model has reached the required refinement state for the intented analysis, a simulative examination can take place.

Depending on the degree of model refinement, the simulation results either evaluate simple dynamic behaviour or serve as performance measure and so as evaluation of more complex behaviour. For example the detection of ressource bottlenecks or of violations of timing limits are common performance analysis goals.

It is mandatory, that all of the simulation results are returned to the model. Especially performance measurements can establish a basis for system analysis on higher hierarchy level during integration of the elaborated model parts. In this paper that kind of model integration is called virtual integration as counterpart to the physical assembly of already realised subsystems, previously named integration.

At the end of a finite number of refinement iterations the system should be modeled adequately detailed. At this point of development the realisation can be executed by appropriate expert teams.

Finally the subsystems can be integrated, verified and validated as described before. But in opposition to the requirements based process, during this assembly task significantly less errors should be discovered. This expectation is motivated due to the prior verification and validation within the virtual model integration.

Figure 2 gives a sketch of the main tasks of a model based systems engineering process as described previously.

As mentioned before, one key advantage of model based development is the increasing accuracy of description. Therefore the risk of misunderstanding and error causing assumptions can be alleviated significantly.

A second key benefit is the possibility of simulation of the model. Thus system verification and validation can take place during the so called virtual integration and errors, even in highly interconnected subsystems, can commonly be avoided or at least discovered early.

However, modelling and simulation require the application of an appropriate modelling language and also an adequate simulation environment supporting that language. Especially while modelling mission level scenarios for example different probabilistic failure models are required.

Furthermore also support of different models of computation are needed for different analysis tasks. For instance discrete event, finite state machine or various data flow computation models are commonly used for analysis of embedded systems.

Besides the need for adequate tool support, also the paradigm change contributes to the previously described lack

of acceptance of model based engineering processes. Not only experienced engineers but also management stakeholders shy away from the increasing efforts this paradigm shift implies.

4. OUTLINE OF A MODEL DRIVEN REQUIREMENTS BASED SYSTEMS ENGINEERING METHOD

As depicted within the prior sections, well-established solely requirements based processes are not able to manage the development of highly interconnected embedded systems. Preservation of consistency is not guaranteed, verification and validation at early development stages are infeasible.

On the other hand, solely model based engineering suffers from a lack of acceptance. Especially the implicitly included process artefacts and the therefore missing baseline documents appear disqualifying in commonly concerned, strongly regulated domains like avionics and automotive.

In the following outline of a systems engineering method the focus will be placed on concepts for combining model and requirements based processes. Neither models nor requirements will be neglected, but rather an appropriate fusion shall be described.

The notion of this paper is, that requirements cannot be omited. This is because there probably always will be stakeholders without knowledge about modeling languages but nonetheless specifying performance and functional parameters of a system.

Hence also the outlined model driven requirements based systems engineering method (MDRBSE) starts with eliciting top level requirements. As already depicted at RBSE, these top level requirements specify the key capabilities and characteristics of the intended system.

In addition to the established development method a significantly higher grade of application of UML (see [8]) and SysML (see [9]) diagrams is proposed. In particular these techniques shall enhance accuracy of interdisciplinary communication. Furthermore the usage of UML, SysML and an appropriate corresponding software tool[2] provides a basis for the later process steps by establishing a comprehensive requirements data model.

Also the subsequent steps of requirements decomposition take place basically similar to RBSE. Within this task the complex system is disassembled top-down first into subsystems and later into elementary functionalities.

In opposition to RBSE the decomposition of the textual requirements can be supported by expanding the previously generated data model. Especially constituting a hierarchical structured requirements tree offers the possibility of strictly guiding decomposition. Thus traceability can be preserved much better and dependencies between requirements are more transparent. Figure 3 shows such a decomposition tree.

It shall be pointed out, that in every node of the requirements decomposition tree additional clarifying UML or SysML diagrams can be annotated. Also the tree itself can be generated using the notation of the SysML requirements diagram. Within this representation the system and the subsystems of figure 3 consist of more than one requirement. Accordingly the number of edges between a system and a subsystem also is higher than displayed. Anyway the tree structure must be kept for optimal traceability.

[2] for an overview of appropriate software tools and further information for example see [11]

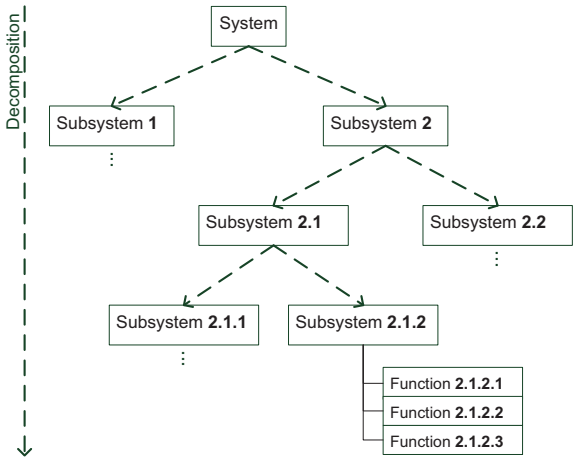

Fig. 3. Requirements decomposition tree

After a finite number of decomposition steps the top level system requirements should be disassembled and thus devided into basic functionalities and nonfunctional requirements. The functional requirements now can be worked out by means of UML and SysML behavioural diagrams like for example activity or state machine diagrams. The nonfunctional requirements stay annotated for later evaluation.

On basis of the elaborated behavioural models the elementary functionalities can now be verified and validated seperated from each other. Assuming an adequate modelling and simulation environment, the UML models can either be executed directly or after translation into an equivalent simulation model[3]. By execution of the UML behaviour model and after comparison of the performance results to the annotated nonfunctional characteristics, the basic functionalities are sufficiently analysed. Any discovered error has to be traced and propagated back into the behavioural and requirements model.

After successful verification and validation of the basic functionalities, integration can take place bottom-up according to the previously generated decomposition tree. This way step by step the basic functional model parts are assembled resulting in hierarchical models of small and later larger subsystems. Thus growing fragments of the requirements tree are covered.

Any assembled subtree can afterwards also get verified and validated by simulation. This way complex behaviour and interaction between different functionalities can be checked for errors. Furthermore non-functional parameters like for example performance requirements can be monitored during simulation. It shall again be pointed out, that any discovered error has to be traced and fixed in the model.

Since the task of assembling basic funtionality in a model is quite similar to the problem of putting together already realised parts, the term integration appears to be appropriate also for the model. To differentiate between model and realised elements, the term virtual integration was chosen for model assembly.

As far as the early detection of errors is concerned, especially regarding the compliance to non-functional performance limits, even the described virtual integration bottom-up is quite late within the entire process. For this reason also

[3] for examples of translation from UML to another, in this case proprietary modelling language see [3] or [12]

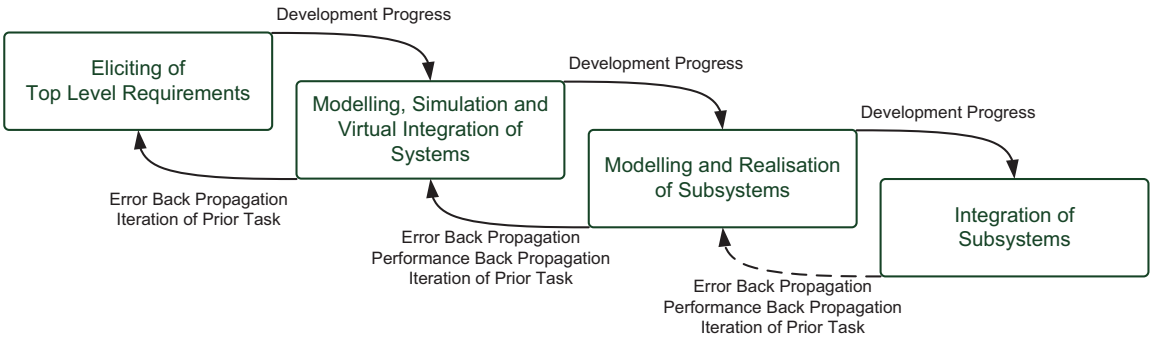

Fig. 4. Main steps of the outlined model driven requirements based systems engineering process

integration of not completely elaborated subtrees is proposed. Assuming the application of an appropriate modelling and simulation environment, this kind of virtual integration top-down can be carried out. During this, black-box model fragments can for instance be parameterised reckoning their compliance to the given performance and behavioural requirements without having the real functionality realised yet.

Applying the proposed top-down virtual integration, the entire system can be verified and validated by simulation at any stage of development. Completely elaborated subsystems can be integrated using their real functionality or using the evaluated performance characteristics. Black-box model parts can be integrated using the described assumptions. Depending on what level of completeness the model has reached, the analysis results can give a more or less precise evidence of the compliance to the specification. Those results should be returned to the model for further processing anyway.

If the entire system is completely elaborated as model and fairly verified and validated, realisation and integration can take place. This tasks again can be carried out quite similar to RBSE. But since the model was executed before, a high level of automation can be expected during realisation of software or hardware logic. Integration is also supposed to be less labour-intensive, because the model was sufficiently checked before.

Figure 4 shows the main steps of the model driven requirements based systems engineering process as described previously.

5. CONCLUSION

The outlined model driven requirements based systems engineering method has a lot of similarity to the solely requirements based process. But it is also characterised by the manifold application of UML and SysML models supporting the interdisciplinary communication, lifting accuracy up to a significantly higher level and guiding the development process itself.

Besides this usage of more or less static model structures also the utilisation of dynamic models is a key feature. Especially the early verification and validation of the systems by virtual integration and subsequent simulative analysis is supposed to enhance the quality of specification substantially.

Summing up, the described development method combines the excellence of requirements and model based engineering. Regardless of the fundamental changes in the used languages, the outlined model driven development should be able to integrate into well-established processes seamlessly because of the high similarity to RBSE concerning the workflow.

Since currently the evaluation of the described notion of engineering method is in progress, a concluding review can not be provided at this point of research.

6. OUTLOOK

As denoted previously, currently some evaluation work is in progress.

On the one hand, some efforts are put into the expansion of the results of [3] and [12]. Especially the translation of more diagram types from UML to a simulation model would be desireable. Also the work of [13] has to be taken into account. In particular the idea of maintaining a generic UML model and generating domain specific working and simulation models out of it seems to be very promising.

On the other hand, the suitability of the outlined method in a real working environment is evaluated currently. A prototype of a requirements centric but model driven engineering process was elaborated including a real workflow and tooling. Although in this prototype process UML was replaced by a more formal and less flexible proprietary modelling language, this evaluation is expected to provide significant results.

7. REFERENCES

[1] Chris Rupp and SOPHIST GROUP, *Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis*, Hanser, 4. edition, 2007.

[2] Horst Salzwedel, "Mission Level Design of Avionics," in *AIAA IEEE DASC 2004 - The 23rd Digital Avionics Systems Conference, 24.-28. October, Salt Lake City*, 2004.

[3] Tommy Baumann and Horst Salzwedel, "Mission Level Design using UML 2.0," in *Net.ObjectDays 2005, 19.-22. September, Erfurt*, 2005.

[4] Torsten Klein, Mirko Conrad, Ines Fey, and Matthias Grochtmann, "Modellbasierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler," in *Modellierung 2004, 23.-26. März, Marburg*. DaimlerChrysler AG, Forschung und Technologie, Methoden und Tools, 2004.

[5] Devamani Ott, Peter Jaeschke, and Rainer Endl, "Requirements Engineering Barometer - Schlussfolgerungen - Auszug aus der Masterarbeit," Tech. Rep., FHS St.

Gallen - Hochschule für Angewandte Wissenschaften, 2008.

[6] Herbert Stachowiak, *Allgemeine Modelltheorie*, Springer, 1973.

[7] Gunar Schorcht, *Integrated Mobile Communication Systems at Mission Level*, Dissertation, Technische Universität Ilmenau, 2000.

[8] OMG - Object Management Group, *OMG Unified Modeling Language (OMG UML), Version 2.3*, Mai 2010.

[9] OMG - Object Management Group, *OMG Systems Modeling Language (OMG SysML), Version 1.2*, Juni 2010.

[10] Knowledge Based Systems Inc., "IDEF Family of Methods - A Structured Approach to Enterprise Modeling & Analysis," www.idef.com.

[11] Martin Albiez, Frederik Nothelfer, and Wolfgang Scherer, "Brauchbare UML-Werkzeuge - Fachstudie Nr. 74," Tech. Rep., Universität Stuttgart, Institut für Softwaretechnologie, Abteilung Software Engineering, 2007.

[12] Tommy Baumann and Horst Salzwedel, "Verwendung von UML-Modellen in einem integrierten Mission Level Design Flow zur Entwicklung von Hard- und Software," in *49. Internationales Wissenschaftliches Kolloquium, 27.-30. September, Ilmenau*, 2004.

[13] Johannes Gross, Axel Reichwein, Stephan Rudolph, Dagmar Bock, and Rene Laufer, "An Executable Unified Product Model Based on UML to Support Satellite Design," in *AIAA Space 2009 Conference and Exposition: 14.-17. September, Pasadena*, 2009.